**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**TRƯƠNG NGỌC KHA**

**HUỲNH ĐĂNG KHOA**

**KHÓA LUẬN TỐT NGHIỆP**

**MỘT SỐ CẢI TIẾN CỦA TÍNH TOÁN CHƯƠNG TRÌNH LOGIC THEO TIẾP CẬN ĐẠI SỐ TUYẾN TÍNH**

**GIẢNG VIÊN HƯỚNG DẪN**

**ThS. NGUYỄN ĐÌNH HIỂN**

**TP. HỒ CHÍ MINH, 2018**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**TRƯƠNG NGỌC KHA – 14520401**

**HUỲNH ĐĂNG KHOA - 14520422**

**KHÓA LUẬN TỐT NGHIỆP**

**MỘT SỐ CẢI TIẾN CỦA TÍNH TOÁN CHƯƠNG TRÌNH LOGIC THEO TIẾP CẬN ĐẠI SỐ TUYẾN TÍNH**

**GIẢNG VIÊN HƯỚNG DẪN**

**ThS. NGUYỄN ĐÌNH HIỂN**

**TP. HỒ CHÍ MINH, 2018**

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

* 1. …………………………………………. – Chủ tịch.
  2. …………………………………………. – Thư ký.
  3. …………………………………………. – Ủy viên.
  4. …………………………………………. – Ủy viên.

MỤC LỤC

[Chương 1. TỔNG QUAN 11](#_Toc518109338)

[1.1. Giới thiệu bài toán 11](#_Toc518109339)

[1.2. Mục tiêu đề tài 12](#_Toc518109340)

[1.3. Phương pháp thực hiện 12](#_Toc518109341)

[Chương 2. TÍNH TOÁN CHƯƠNG TRÌNH LOGIC THEO TIẾP CẬN ĐẠI SỐ 13](#_Toc518109342)

[2.1. Cơ sở lý thuyết 13](#_Toc518109343)

[2.2. Chương trình logic dạng Horn 13](#_Toc518109344)

[2.2.1. Biểu diễn ma trận cho chương trình horn 13](#_Toc518109345)

[2.2.2. Thuật toán để tìm mô hình tối tiểu của chương trình logic dạng Horn 18](#_Toc518109346)

[2.3. Chương trình logic dạng tuyển 19](#_Toc518109347)

[2.3.1. Tensor biểu diễn chương trình logic dạng tuyển 19](#_Toc518109348)

[2.3.2. Thuật toán để tìm mô hình tối thiểu của một chương trình dạng tuyển 24](#_Toc518109349)

[Chương 3. MỘT SỐ PHƯƠNG PHÁP CÁI TIẾN 27](#_Toc518109350)

[3.1. Chương trình xác định (Define Program) 27](#_Toc518109351)

[3.1.1. Chương trình SD 27](#_Toc518109352)

[3.1.2. Non-SD program 28](#_Toc518109353)

[3.1.3. Thuật toán tìm mô hình tối tiểu 29](#_Toc518109354)

[3.1.4. Tính bằng ma trận con 30](#_Toc518109355)

[3.2. Chương trình dạng chuẩn (Normal Program) 33](#_Toc518109356)

[3.2.1. Tính toán mô hình ổn định 33](#_Toc518109357)

[3.2.2. Thuật toán để tính toán mô hình ổn định 34](#_Toc518109358)

[3.2.3. Tính toán bằng ma trận con 36](#_Toc518109359)

[Chương 4. KẾT QUẢ THỬ NGHIỆM 38](#_Toc518109360)

[4.1. So sánh với chương trình logic dạng horn 38](#_Toc518109361)

[4.2. So sánh với chương trình logic dạng normal program 39](#_Toc518109362)

[Chương 5. KẾT LUẬN 40](#_Toc518109363)

[5.1. Chủ đề cấp độ 2 40](#_Toc518109364)

[5.1.1. Chủ đề cấp độ 3 40](#_Toc518109365)

[5.2. Chủ đề cấp độ 2 40](#_Toc518109366)

DANH MỤC HÌNH VẼ

[Hình 1.1: Tên hình 1 3](#_Toc367742554)

DANH MỤC BẢNG

[Bảng 1.1: Tên bảng 1 3](#_Toc367742567)

[Bảng 2.1: Tên bảng 1 4](#_Toc367742568)

DANH MỤC TỪ VIẾT TẮT

TÓM TẮT KHÓA LUẬN

MỞ ĐẦU

# TỔNG QUAN

## Giới thiệu bài toán

Liệu ta có thể biến những vấn đề tính toán khó khăn trở nên dễ dàng ? Đó chắc hẳn đang là câu hỏi đang được nhiều người đặt ra. Nếu chúng ta giới hạn phạm vi của câu hỏi trong việc giải quyết các vấn đề tính toán trên máy tính và nếu chúng ta định nghĩa “dễ dàng” ở đây có nghĩa là sử dụng một mô hình đơn giản và trực quan cho việc tính toán thì khi đó, Answer Set Programming (ASP) sẽ là câu trả lời phù hợp cho các điều kiện nêu trên.

ASP là một dạng lập trình hướng tới các vấn đề gặp khó khăn trong việc thực hiện cong việc tìm kiếm. Nó dựa trên mô hình ổn định của lập trình logic. Trong ASP, các vấn đề tìm kiếm sẽ được giảm xuống để tính toán mô hình ổn định và các answer set solver – chương trình tạo ra mô hình ổn định – được sử dụng để thực hiện tìm kiếm. Về mặt nguyên tắt, quá trình tính toán bằng phương pháp này luôn kết thúc (không giống như truy vấn Prolog, có thể dẫn đến lặp vô hạn).  
ASP mới chỉ được biết đến trong khoảng vừa hơn một thập kỉ qua.Tuy nhiên, nguồn gốc của nó đã có từ rất lâu về trước. Nó là kết quả của cả một quá trình nghiên cứu trong lĩnh vực biễu diễn tri thức lập trình logic cũng như các công cụ tính toán và các định nghĩa dùng để biểu diễn rõ ràng các vấn đề trong ngôn ngũ lập trình. ASP dựa trên các lĩnh vực nghiên cứu trên, tập trung đến việc cân bằng giữa khả năng biễu diễn các vấn đề logic, dễ sử dụng và hiệu quả trong tính toán. Hiệu quả của ASP có thể thấy được qua việc nó đã được ứng dụng trong các lĩnh vực như: sinh học phân tử, hệ thống hỗ trợ quyết định cho tàu con thoi, hệ thống quản lý lực lượng lao động cho cảng biển Gioia Tauro, Reggio Calabria, Italy.

## Mục tiêu đề tài

Về cơ bản, mục tiêu của đề tài là nghiên cứu phương pháp tính toán ngữ nghĩa của một chương trình logic dựa trên đại số tuyến tính. Đồng thời thực hiện một số cải tiến đối với các thuật toán này để việc tính toán trở nên hiệu quả hơn. Các chương trình logic được biểu diễn bằng ma trận và mô hình tối tiểu của nó sẽ được xác định dựa trên các kỹ thuật tính toán trên ma trận. Bên cạnh đó, các chương trình logic dạng tuyển cũng được nghiên cứu và biểu diễn bằng các tensor. Thông qua đó, sử dụng các tensor này để tính toán các mô hình tối tiểu của chương trình logic dạng tuyển. Hơn nữa, các kỹ thuật tối ưu cho quá trình tính toán các mô hình tối tiểu cũng được nghiên cứu và đề xuất. Kết quả phân tích và thử nghiệm trong thực tế cho thấy phương pháp tối ưu được đề xuất mang lại những hiệu quả nhất định.

## Phương pháp thực hiện

# TÍNH TOÁN CHƯƠNG TRÌNH LOGIC THEO TIẾP CẬN ĐẠI SỐ

## Cơ sở lý thuyết

Các định nghĩa trong mục này được trích từ [8, 10]. Tensor bậc *k* là kết quả của phép tính tích tensor của *k* vector. *Tensor bậc 1* là một vector *v* ∈ *n*, *tensor bậc 2* là một ma trận *M* ∈ *m*×*n*, *tensor bậc 3* là một mảng ba chiều *T* ∈ *m*×*n*×*p* (*m*,*n*,*p*≥1). *Lớp* của một tensor bậc *k* là một mảng hai chiều của một tensor. Tensor bậc ba *T* ∈ *m*×*n*×*p*gồm các lớp bằng cách cố định một giá trị *j* (1≤ *j* ≤ *p*). Lớp thứ *j* của một tensor bậc 3 *T*là một ma trận và được ký hiệi là *T*::*j*.

*Tích* (2-mode) của một tensor *T* ∈ *m*×*n*×*p* và vector *v* ∈ *n* được ký hiệu là *T* • *v*. Kết quả của tích này là một ma trận *m* × *p* có phần tử là .

Khi *M* ∈ *m*×*n* là một ma trận và *v* ∈ *n*, tích *M* • *v* sẽ là một vector trong *m*.

## Chương trình logic dạng Horn

Xét ngôn ngữ L gồm có chứa một tập hữu hạn các biến mệnh đề và các phép toán logic ¬, ∧, ∨ và ←. L có các ký hiệu và ⊥ đại diện cho các giá trị *true* và *false* một cách tương ứng. Cho một chương trình logic *P*, tập tất cả các biến mệnh đề xuất hiện trong *P* được gọi là cơ sở Herband của P, ký hiệu *BP*. Giả định rằng {, ⊥} ⊂ *BP*.

### Biểu diễn ma trận cho chương trình horn

Một chương trình logic *P* được là một chương trình (logic) *dạng* *Horn* khi và chỉ khi *P* là một tập hữu hạn các luật có dạng:

 (1)

Trong đó, *h* và *bi* là các biến mệnh đề. Luật (1) là một *sự kiện* (*fact*)nếu “*h*←” và (1) là một *ràng buộc* (constraint) nếu “”. Khi đó, ta có thể viết lại chúng dưới dạng “*h* ←” và “← *b*1∧…∧*bm*”. Một chương trình được gọi là *xác định* (definite program) khi nó không chứa các ràng buộc.

Với luật *r* có dạng (1) trong chương trình dạng Horn *P*, đặt:

*head*(*r*) = *h* và *body*(*r*) = {*b*1*,…, bm*}.

Xét một tập *I* thỏa {} ⊆ *I* ⊆ *BP*, ⊥∉ *I*.

Khi đó, *I* được gọi là một *mô hình* của *P* nếu {*b*1*,…, bm*}⊆ *I* suy ra *h* ∈ *I* với mọi luật có dạng (1) trong *P*. Một mô hình *I* là *mô hình tối tiểu* (*least model*)của *P* nếu *I* ⊆ *J* ∀ *J* là mô hình của *P*.

Ánh xạ được định nghĩa như sau:

*TP*(*I*) = {*h* | *h* ← *b1* ∧ …∧ *bm* ∈ *P* và {*b1*,…,*bm*}⊆ *I*}

Luỹ thừa của *TP* được định nghĩa như sau:

 (*k* ≥ 0).

Cho *I* ⊆ *BP*, khi đó sẽ có một điểm cố định.

Cho một chương trình *P* xác định, điểm cố định cũng chính là mô hình tối tiểu của *P*.

**Định lý 2.2.1 [9]:** Cho chương trình Horn P và ánh xạ *TP*.

1. Đặt , với *J* là mô hình của *P*. Thì *M* là mô hình tối tiểu của *P*.
2. *M* là điểm cố định nhỏ nhất của *TP*, nghĩa là *M =* .

**Định nghĩa 2.2.1:** (*điều kiện* -*MD*)

Chương trình dạng Horn *P* được gọi là thỏa *điều kiện-MD* khi và chỉ khi *P* thỏa điều kiện sau:

Với mọi luật *r1* và *r2* tthuộc *P* (*r1* ≠ *r2*):

*head*(*r1*) = *head*(*r2*) suy ra |*body*(*r1*)| ≤ 1 và |*body*(*r2*)| ≤ 1.

(gọi là *điều kiện-MD* (*multiple definitions*)).

**Bổ đề 2.2.1:** Cho chương trình dạng Horn *P* và *M* là mô hình tối tiểu của *P*. Khi đó, có một chương trình dạng Horn *P’* thoả điều kiện-MD, đồng thời:

*P’* có mô hình tối tiểu là *M’* và *M’* ∩ *BP* = *M*.

*Chứng minh.*  Đặt *S* := {*r* | *r* ∈ *P*, |*body*(*r*)| > 1}.

*P’* = *P*; *BP’* = *BP* ;

Với mỗi *r* = *hr* ← *br*1∧ …∧ *brm* ∈ *S*, tạo một biến mệnh đề mới *cr* ∉ BP và hai luật mới: *hr* ← *cr* và *cr* ← *br*1∧ …∧ *brm*.

*BP’* := *BP* ∪ {*cr*}.

*P’*:= *P’* \ {*r*} ∪ {*hr* ← *cr* , cr ← *br*1∧ …∧ *brm*}.

Khi đó *P’* sẽ thoả điều kiện-MD và mô hình tối tiểu *M’* của *P’* thoả: *M’* ∩ *BP* = *M*.

**Định nghĩa 2.2.2 [8]:** *(ma trận biểu diễn chương trình Horn)*

Cho một chương trình dạng Horn *P* thoả điều kiện-MD và *BP* = {*b*1*,…, bn*}. Khi đó, *P* được biểu diễn bởi một ma trận *MP* ∈ *n*×*n* với mỗi phần tử *aij* (1 *≤* *i*, *j* *≤* *n*) trong *MP* có giá trị:

1. *aij* = 1 nếu *bi* = hoặc *bj* = ⊥.

 nếu có trong P.

3. Các trường hợp khác, *aij* = 0.

Đặt row*i*(*MP*) = *(ai*1*, ai*2*, …, ain)* ∈ n,

col*j*(*MP*) = *(a*1*j, a*2*j, …, anj)*T(1 *≤* *i*, *j* *≤* *n*)

(*v*T là một chuyển vị của vector *v* ∈ n).

**Định nghĩa 2.2.3** **[8]**: (*vector biểu diễn mô hình*)

Cho *P* là một chương trình dạng Horn và *BP* = {*b*1*,…, bn*}. thì mô hình *I* của *P* được biểu diễn bởi một vector *v*= (*a*1*, . . . , an*)T trong đó mỗi phần tử *ai* biểu diễn giá trị thực của mệnh đề *pi* sao cho *ai* = 1 nếu *bi* ∈ *I* (1 *≤ i ≤ n*); và *ai* = 0 trong các trường hợp còn lại.

Vector biểu diễn cho *I* = {} is được ký hiệu là *v*o.

Giả sử hai vector *v*= (*a*1*,…, an*)T và *w*= (*b*1*,…,bn*)T lần lượt biễu diễn các mô hình *I* ⊆ *BP* và *J* ⊆ *BP* tương ứng.

Khi đó *v* ≤ *w* nếu *ai* ≤ *bi* ∀ 1 ≤ *i* ≤ *n*.

**Ví dụ 2.2.1:**

a) Xét *P1* = {*p* ← *q*, *p* ← *r*, *q* ← *r* ∧ *s*, *r* ← , ← *q*} với *BP1* = {*p*, *q*, *r*, *s*, , ⊥} thì *MP1* ∈ 6×6 là ma trận vuông.

 Hàng 1 “011001” tượng trưng *p* ← *q*, *p* ← *r*, *p* ← ⊥.

Hàng 2 “00½ ½01” biểu diễn các luật *q* ← *r* ∧ *s*, *q* ← ⊥.

Hàng 3 “000011” biểu diễn các luật *r* ←, *r* ← ⊥.

Hàng 4 “000001” biểu diễn luật *s* ← ⊥.

Hàng 5 “111111” biểu diễn các luật ← *p*, ← *q*, ← *r*, ← *s*, ← , ← ⊥.

Hàng 6 “010001” biểu diễn các luật ⊥ ← q, ⊥ ← ⊥.

b) Xét *P2* = {*p* ← *t*, *t* ← *q* ∧ *r*, *p* ← *u*, *u* ← *s* ∧ *q*, *s* ← *r*, *r* ← *q*} với *BP2* = {*p*, *q*, *r*, *s*, *t*, *u*} thì *MP2* ∈ 6×6 là ma trận (vuông).

 Để biểu diễn đơn giản hơn, trong ví dụ này chúng ta bỏ qua và ⊥.

Hàng 1 “000011” biểu diễn các luật *p* ← *t*, *p* ← *u*.

Hàng 3 “010000” biểu diễn luật *r* ← *q*.

Hàng 4 “001000” biểu diễn luật *s* ← *r*.

Hàng 5 “0 ½ ½ 000” biểu diễn luật *t* ← *q* ∧ *r*.

Hàng 6 “0 ½ 0 ½00” biểu diễn luật u ← *s* ∧ *q*.

Giả sử *MP* ∈ *n*×*n* là ma trận biểu diễn cho một chương trình dạng Horn *P*. Gọi *v* ∈ n là vector biểu diễn cho một mô hình *I* ⊆ *BP*, khi đó *MP*• *v*= (*a*1*,…, an*)T. Biến đổi *MP* • *v* thành một vector *w*= (*a*1*’,…, an’* )T trong đó *ai’* = 1 (1 ≤ *i* ≤ *n*) nếu *ai* ≥ 1; và *ai’* = 0 nếu *ai* < 1. Ký hiệu: *w* = *θ(MPv)*.

**Định lý 2.2.2 [8]**: Cho *P* là một chương trình xác định thoả điều kiện-MD và *P* có ma trận biểu diển là *MP* ∈ n×n. Gọi *v* ∈ n là vector biểu diễn cho một mô hình *I* ⊆ *BP*.

Khi đó, *w* ∈ n là một vector biểu diễn cho *J = TP(I)* khi và chỉ khi *w = θ(MPv).*

*Chứng minh*.

Cho và *v =* khi đó *w = θ(MPv) =*  với *xk=ak1y1+…+aknyn*

+ Giả sử *w = θ(MPv) =* với *xk’ = 1xk1.*

Ta có: *xk = ak1y1+…+aknyn1*

Nếu *rawj­(Mp) = T* thì *akj = 1*.

Cho *{b1,…,bm}{ak1,…,akn}* sao cho bi 0. Ta có 2 trường hợp:

i/ *bi = 1 (1)* và *b1yj1 +…+ bmyjm1*

Khi có tồn tại luật: *pkpi* trong P sao cho *pi = colj(Mp)* để *bi = akj (1)* và *{p1,…,pm}I* nghĩa là *pkTp(I)* khi *pk = rawk(w).*

ii/ *bi = 1/m (1)* và *b1yj1 +…+ bmyjm1*

Khi có tồn tại luật: *pkp1 pm* trong P sao cho *pi = colj(Mp)* để *bi = akj (1)* và *{p1,…,pm}I* nghĩa là *pkTp(I)* khi *pk = rawk(w).*

+ Giả sử *J = Tp(I):*

Định nghĩa *w=* biểu diễn *J = Tp(I).*

Ta có: *w = θ(MPv) =* là một vector sao cho:

*xk 1 rawk (w = θ(MPv)) = T* hoặc *rawk ( w = θ(MPv) ) Tp(I)*

Khi đó w sao cho: *xk’ = 11* in *w* = *θ(MPv)*

Vì vậy *w* = *θ(MPv)*

Cho ma trận biểu diễn *MP* ∈ n×n và một vector *v* ∈ n. Định nghĩa:

*θ(MPvk+1)* = *θ(MP*(*MPvk*) và *θ(MPv1)* = *θ(MPv)* (*k* ≥ 1).

Khi *θ(MPvk+1)* = *θ(MPvk*)*với* *k* ≥ 1, đặt: *FP*(*MPv*) = *θ(MPvk)*.

Theo [8], ta có: *m* ∈ n là một vector biểu diễn cho mô hình tối tiểu cua *P* khi và chỉ khi *m* = *FP*(*MPv*o) với *v*o là vector biểu diễn *I* = {}.

### Thuật toán để tìm mô hình tối tiểu của chương trình logic dạng Horn

Dựa trên định lý 3.2, chúng ta có thuật toán để tìm mô hình tối tiểu của một chương trình *P* thoả mãn điều kiện-MD. Không mất tính tổng quát, ta chỉ cần xét *P* là một chương trình xác định (không chứa các ràng buộc):

**Input:** *P* là một chương trình xác định thoả mãn điều kiện-MD và *F* là một tập các sự kiện trong *P.*

**Output:** Vector *u* biểu diễn mô hình tối tiểu của *P*.

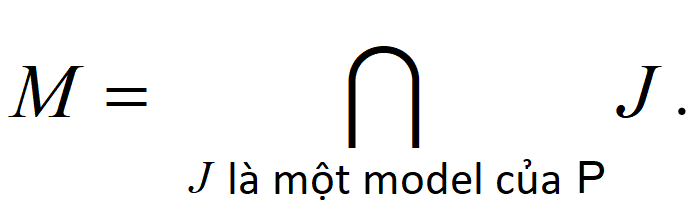
**Thuật toán 2.2.1:**

|  |  |
| --- | --- |
| **Bước 1:** *Tạo ma trận*  *MP = (aij)*1≤*i,j*≤*n* *biểu diễn chương trình P*  for *i* from 1 to *n* do  for (luật *r* ∈ *P*) do  if *bi* ∈ *head*(*r*) then  *m* := 1/|*body*(*r*)|  for *j* from 1 to *n* do  if bj ∈ *body*(*r*) then  *aij* = *m*  end do; # *r* ∈ *P*  end do;  **Bước 2:** *Tạo vector*  *vo = (v1,…vn) biểu diễn tập F*  for *i* from 1 to *n* do  if *bi* ∈ *F* then *vi* = 1  else *vi* = 0  end do; | **Bước 3:** *Mô hình tối tiểu của P*  *v*:= *vo*  *u*:=*MP* • *v*  *u*:= *u* + *vo*  for *i* from 1 to *n* do  #Biến đổi *u* = *θ(MPv)*  if *u*[*i*] ≥ 1 then *u*[*i*] := 1  else *u*[*i*]:=0;  end do;  while *u ≠ v* do  *v*:= *vo*  *u*:=*MP* • *v*  *u*:= *u* + *vo*  for *i* from 1 to n do  #Biến đổi *u* = *θ(MPv)*  if *u*[*i*] ≥ 1 then *u*[*i*] := 1  else *u*[*i*]:=0;  end do; #while  return *u*; |

**Định lý 2.2.3:** Cho P là một chương trình xác định và ánh xạ TP. Gọi  sao cho:



Đặt

.

Khi đó, ta có: a) ∀ *k* ≥ 0, *u*k ⊆ *TP*(*u*k) = *u*k+1 .

b).

c) .

*Chứng minh*.

a) Ta có thể dễ dàng chứng minh bằng phương pháp quy nạp.

b) Từ (a) ta có: *u*k ⊆ *TP*(*u*k) = *u*k+1 ∀ *k* ≥ 0,

và BP có *n* phần tử

 và *no*≤ *n*.

c) Kết quả này được suy ra từ định lý 2.2.1.

Trong thuật toán 3.1, độ phức tạp của θ(*MPv*) là **O(*n2*)**. Theo định lý 2.2.3, số lần thực hiện θ(*MPv*)nhiều nhất là ***n*** lần. Như vậy, độ phức tập của bước 3 là **O(*n3*)**.

## Chương trình logic dạng tuyển

### Tensor biểu diễn chương trình logic dạng tuyển

Một chương trình logic *P* là một *chương trình* (*logic*) *dạng tuyển (disjunctive (logic) program)* khi và chỉ khi *P* là một tập hữa hạn các luật có dạng:

*h*1∨…∨*hl* ← *b*1∧…∧*bm* (*l* ≥ 1, *m* ≥ 0) (2)

Trong đó *hi* và *bj* là các biến mệnh đề. Luật dạng (2) được gọi là một *luật tuyển* nếu *l* > 1, là *sự kiện* nếu vế phải của (2) là và gọi là một *ràng buộc* nếu vế trái của (2) là ⊥. Khi đó, một sự kiện dạng tuyển được viết đơn giản như sau “*h*1∨…∨ *h­l* ← ”.

Cho luật *r* trong một chương trình dạng tuyển *P*, đặt:

*head*(*r*) = {*h*1 *, … , h­l*} và *body*(*r*) = {*b*1*,…, bm*}.

Một tập *I* thỏa điều kiện {} ⊆ *I* ⊆ *BP* được gọi là một mô hìnhcủa một chương trình dạng tuyển *P* nếu *body*(*r*) ⊆ *I* suy ra *head*(*r*) ∩ *I* ≠ ∅ ∀ *r* ∈ *P* và ⊥ ∉ *I*.

Một mô hình *I* là một mô hình tối thiểu (minimal model)của *P* nếu không tồn tại mô hình *J* của *P* mà *J* ⊂ *I*.

**Định nghĩa 2.3.1:** (*điều kiện-MD cho chương trình dạng tuyển*)

Chương trình dạng tuyển *P* được gọi là thỏa Điều kiện-MD nếu nó thỏa những điều kiện sau:

∀ *r1* ∈ *P*, *r2* ∈ *P*, *r1* ≠ *r2*:

*head*(*r1*)∩*head*(*r2*) ≠ ∅ suy ra |*body*(*r1*)|≤1 và |*body*(*r2*)|≤ 1.

(gọi là *điều kiện-MD* (*multiple definitions*)).

Định nghĩa này trùng với định nghĩa 2.2.1 nếu *P* là một chương trình logic dạng Horn.

**Bổ đề 2.3.1:** Cho một chương trình logic dạng tuyển *P*, khi đó, có một chương trình logic dạng tuyển *P’* thoả điều kiện-MD, đồng thời:

P có một mô hình tối thiểu M khi và chỉ khi *P’* có mô hình tối thiểu là *M’* sao cho *M’* ∩ *BP* = *M*.

*Chứng minh 1.* Đặt *S* := {*r* | *r* ∈ *P*, |*body*(*r*)| > 1}.

*P’* := *P*; *BP’* := *BP* ;

Với mỗi *r* = *hr*1∨…∨ *hr­l* ← *br*1∧ …∧ *brm* ∈ *S* (*l* ≥ 1, *m* > 1), taọ một biến mới c*r* ∉ BP và hai luật mới: *hr*1∨…∨ *hr­l* ← *cr* và *cr* ← *br*1∧ …∧ *brm*.

*BP*’ := *BP* ∪ {*cr*}.

*P’*:= *P’* \ {*r*} ∪ { *hr*1∨…∨ *hr­l* ← *cr* , cr ← *br*1∧ …∧ *brm*}.

Khi đó *P’* thỏa điều kiện-MD, đồng thời:

P có một mô hình tối thiểu M khi và chỉ khi *P’* có mô hình tối thiểu là *M’* sao cho *M’* ∩ *BP* = *M*.

Cho *P* là một chương trình dạng tuyển, *chương trình* *tách* (*split program*)của *P* là một chương trình Horn được sinh ra từ *P* bằng cách thay thế mỗi luật tuyển dạng (2) trong *P* bằng một luật Horn:



Theo định nghĩa, *P* có nhiều chương trình tách. Khi *P* có *k* chương trình split, chúng sẽ được ký hiệu là *SP1*,…, *SPk*.

Gọi T là tập các mô hình, khi đó phần tử tối thiểu của tập T được định nghĩa như sau:

min(T) = { *I* | *I* ∈ T và không tồn tại *J* ∈ T sao cho *J* ⊂ *I* }.

**Mệnh đề 2.3.1 [8]**: Cho *P* là một chương trình dạng tuyển và *SP1*,…, *SPk* là các chương trình tách từ P. Gọi *LM* là tập hợp các mô hình tối tiểu (least model) của *SP1*,…, *SPk*. Khi đó min(*LM*) trùng với tập *MM* là tập các mô hình tối thiểu (minimal model) của *P*.

Chứng minh:

Cho M là mô hình tối tiểu của chương trình tách SPj.

Khi đó với mỗi luật r: trong SPj, tồn tai một luật r’:  trong P sao cho head(r’). Vì M thoả r, nên M thoả r’. Vậy, M là một mô hình của P.

Khi đó tâp tối thiểu của M trong min(LM) là một mô hình tối thiểu (minimal model) của P.

Như vậy, min(LM)MM.

Ngược lại, cho MMM.

Khi đó với mỗi luật r:  trong P, {*b*1*,…, bm*} M hàm ý M với i (1 ≤ i ≤ l).

Trong trường hợp này, tồn tại một chương trình tách SPj của P trong đó r được thay bằng .

Khi đó M là mô hình tối tiểu của SPj

Vì M là một mô hình tối thiểu trong LM, MM min(LM).

**Định nghĩa 2.3.2:** (*tensor biểu diễn chương trình* *logic dạng tuyển*)

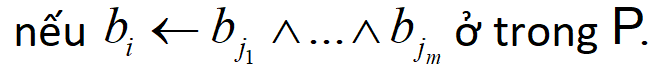
Cho *P* là một chương trình logic dạng tuyển thỏa điều kiện-MD. Giả sử rằng *P* được tách thành các chương trình dạng Horn *SP1*,…, *SPk* (*k* ≥ 1) và *BP* = {*b*1*,…, bn*}. Khi đó *P* được biễu diễn by một tensor bậc ba *UP* ∈ *n*×*n*×*k* như sau:

1. Mỗi lớp *U*::*h* (1 ≤ *h* ≤ *k*) của *UP* là một ma trận *Mh* ∈ *n*×*n* biểu diễn chương trình Horn *SPh*.

2. Mỗi ma trận *Mh* ∈ *n*×*n* có một phẩn tử *aij* (1 ≤ *i*, *j* ≤ *n*):

a) *aij* = 1 nếu *bi* = hoặc *bj* = ⊥





c) Các trường hợp còn lại, *aij* = 0.

**Ví dụ 2.3.1:** Xét chương trình dạng tuyển *P* như sau: *P* = {*p* ∨ *r* ← *s*, *q* ∨ *r* ←, *s* ←}, *BP* = {*p*, *q*, *r*, *s*, }, khi đó *P* được tách thành 4 chương trình dạng Horn:

*SP1* = {*p* ← *s*, *q* ←, *s* ←},

*SP2* = {*p* ← *s*, *r* ←, *s* ←},

*SP3* = {*r* ← *s*, *q* ←, *s* ←},

*SP4* = {*r* ← *s*, *r* ←, *s* ←}.

Chương trình *P* được biểu diễn bởi tensor bậc ba *UP* ∈ 5×5×4 :





Gọi *UP* ∈ *n*×*n*×*k*là một tensor bậc ba, *v* ∈ *n* là vector biểu diễn mô hình *I* ⊆ *BP*, tích (2-mode) *UP* • *v* là một ma trận (*aij*) ∈ *n*×*k*. Ta chuyển đổi *UP* • *v* thành một ma trận *W* = (*a’ij*) ∈  *n*×*k* với *a’ij* = 1 (1 ≤ *i* ≤ *n*, 1 ≤ *j* ≤ *k*) nếu *aij* ≥ 1; và *a’ij* = 0 nếu *aij* < 1. Khi đó, ta ký hiệu *W* = θ(*UP* *v)*.

Cho một tensor *UP* ∈ *n*×*n*×*k* và vector *v* ∈ *n*, định nghĩa:

θ(*UPvm+1)* = θ(*UP*(*UPvm*)) và θ(*UPv1)* = θ(*UPv0)* (*m* ≥ 1).

Khi θ(*UPvm+1)* = θ(*UPvm)* với *m* ≥ 1, đặt: *FP*(*UPv*) = θ(*UPvm)*.

**Định lý 2.3.1 [8]**: Cho *P* là một chương trình dạng tuyển không thỏa điều kiện-MD và *UP* ∈ ∈ *n*×*n*×*k* là tensor biểu diễn cho chương trình *P*. Đặt *MP*= *FP*(*UP*  *v*o) và *vo* là vector biểu diễn I = {}.

Khi đó, mỗi cột vector *m* ∈ n trong *MP*biểu diễn một mô hình tối thiểu của *P* khi và chỉ khi *m*vector nhỏ nhất trong tất cả các cột vector trong *MP* (quan hệ thứ tự giữa các vector đã được định nghĩa trong định nghĩa 2.2.3).

### Thuật toán để tìm mô hình tối thiểu của một chương trình dạng tuyển

**Input:** *P* là một chương trình dạng tuyển thoả mãn điều kiện-MD và *F* là một tập các sự kiện trong *P*. Giả sử rằng *P* không có các ràng buộc.

**Output:** Tập *MM* các mô hình tối thiểu của *P*.

**Thuật toán 2.3.1:**

|  |  |
| --- | --- |
| **Bước 1:** Tách chương trình *P* thành *k* chương trình Horn: *SP1*, *SP2*,…, *SPk*  **Bước 2:** *Tạo tensor bậc ba UP* ∈ *n*×*n*×*k* *để biểu diễn P*  for *h* from 1 to *k* do    for *i* from 1 to *n* do  for (luật *r* ∈ *P*) do  if *bi* ∈ *head*(*r*) then  *m* := 1/|*body*(*r*)|  for *j* from 1 to *n* do  if *bj* ∈ *body*(*r*) then  *a*ij = *m*  end if; # *bi* ∈ *head*(*r*)  end do; # *r* in P  end do;  end do;  **Bước 3:** *Tìm ma trận M* ∈ *n×k trong đó các cột của nó biễu diễn mô hình tối tiểu của SPh (h =*1*…k)*  3.1 Tạo vector *v*F = (*v1*,…*vn*) biễu diễn tập sự kiện *F*.  3.2 Tạo ma trận *Mo* ∈ *n*×*k* với coli(*Mo*) =*v*F (*i* = 1…*k*)  3.3 *M* :*=UP* • *v*F  *M* := *M* + *Mo*  *M’* := *M* | for *i* from 1 to *n* do  # Biến đổi *M’* = θ(*UPv*F)  for *j* from 1 to *k* do  if *M’*[*i*, *j*] ≥ 1 then  *M’*[*i*, *j*] := 1  else *M’*[*i*, *j*] := 0;  end do;  while *M* ≠ *M’* do  *M*:= *M’*  *M’* := *UP* • *M*  *M’* := *M* + *M’*  for *i* from 1 to *n* do  #Biến đổi *M’* = θ(*UPM)*  for *j* from 1 to *k* do  if *M’*[*i*, *j*] ≥ 1 then  *M’*[*i*, *j*] := 1  else *M’*[*i*, *j*] := 0;  end do;  end do; #while  **Bước 4:** *Tìm tập MM các vector biễu diễn các mô hình tối thiểu của P*  *MM* := ∅;  for *i* from 1 to *k* do  *vi :=* col*i*(*M*)*;*  for *j* from 1 to *k* do  *v*j := colj(*M*);  if (*vj* ≤ *vi*) and (*vj ≠ vi*) then break;  end do;  if *j* > *k* then  *MM* := *MM* ∪ {*vi*};  end do;  return *MM*; |

Trong thuật toán 2.3.1, độ phức tạp của phép tính *UP**M* là **O(*k*2×*n*2)**. Theo định lý 2.2.3, số lần thực hiện θ(*UPM)* nhiều nhất là ***n*** lần. Như vậy, độ phức tạp của bước 2.2.3 là **(*k*2×*n*3)**.

# MỘT SỐ PHƯƠNG PHÁP CÁI TIẾN

## Chương trình xác định (Define Program)

Xét ngôn ngữ L gồm có chứa một tập hữu hạn các biến mệnh đề và các phép toán logic ¬, ∧, ∨ và ←. Cho một chương trình logic *P*, tập tất cả các biến mệnh đề xuất hiện trong *P* được gọi là cơ sở Herband của P, ký hiệu *BP*.

Về cơ bản, chương trình xác định được biễu diễn như một chương trình logic dạng Horn.Với một chương trình xác định P ta cũng sẽ tìm được một điểm cố định  cũng chính là mô hình tối tiểu của *P*.

### Chương trình SD

Đầu tiên chúng ta xét một lớp con của chương trình xác định, được gọi là SD-programs.

**Định nghĩa 3.1.1: (SD-program)** một chương trình *P* xác định được gọi là *singly defined* (hoặc SD-*program*) nếu *head*(*r1*) ≠ *head*(*r2*) với mọi luật *r1* và *r2* thuộc *P* (*r1* ≠ *r2*).

**Định nghĩa 3.1.2** **[8]**: **(vector biểu diễn)** Cho *P* là một chương trình xác định và *BP* = {*p*1*,…, pn*}. Thì mô hình *I* của *P* được biểu diễn bởi một vector *v*= (*a*1*, . . . , an*)T trong đó mỗi phần tử *ai* biểu diễn giá trị thực của mệnh đề *pi* sao cho *ai* = 1 nếu *pi* ∈ *I* (1 *≤ i ≤ n*); các trường hợp khác, *ai* = 0. Chúng ta biết row*i*(*v*) = *pi*. Xét *v*= (*a*1*,…, an*)T∈ *n*, *v*[*i*] là phần tử thứ *i*th của *v* (1*≤ i ≤ n*) và *v*[1…*k*] là một vector (*a1,…, ak*)T∈*k* (*k* ≤ *n*).

**Định nghĩa 3.1.3 [8]**: **(hàm ngưỡng)** xét một vector *v*= (*a*1*,…, an*)T ∈ *n*, định nghĩa *θ* (*v*)= (*a*1*’,…, an’* )T khi *ai’* = 1 (1 ≤ *i* ≤ *n*) nếu *ai* ≥ 1; các trường hợp khác, *ai’* = 0. Chúng ta gọi là *hàm-ngưỡng-θ* của *v*.

**Định nghĩa 3.1.4: (ma trận biểu diễn của SD-program)**[[1]](#footnote-1)cho *P* là một SD-program và *BP* = {*p*1*,…, pn*}. Khi đó, *P* được biểu diễn bởi một ma trận *MP* ∈ *n*×*n* với mỗi phần tử *aij* (1 *≤* *i*, *j* *≤* *n*) trong *MP* có giá trị:



2. *aii* = 1 nếu *pi* ← có trong *P*.

3. *aij* = 0, các trước hợp khác.

*MP*được gọi là một *ma trận biểu diễn*. chúng ta viết row*i*(*MP*)= *pi* và col*j*(*MP*) = *pj* (1 *≤* *i*, *j* *≤* *n*).

**Định nghĩa 3.1.5**: **(vector khởi tạo)** cho *P* là một chương trình xác định và *BP* = {*p*1*,…, pn*}. ta có *initial vector* *vo*= (*a*1*, . . . , an*)T vậy *ai* = 1 nếu row*i*(*vo*) = *pi* và sự kiện *pi* ← trong *P*; trường hợp khác, *ai* = 0.

Cho một ma trận biểu diễn *MP* ∈ *n*×*n* và một vector khởi tạo *vo* ∈ *n*, xác định:

*v1* = *θ* (*MPvo*) và *vk*+1 = *θ* (*MPvk*) (*k* ≥ 1).

Tồn tại *vk*+1 = *vk* với *k* ≥ 1. Khi *vk*+1 = *vk*, ta viết: *vk* = *FP*(*MPvo*).

**Định lý 3.1.1**: Cho *P* là một chương trình-SD và *MP* ∈ *n*×*n* là ma trận biểu diễn. ta có *m* ∈ *n* là một vector biểu diễn mô hình tối tiểu của *P* khi và chỉ khi *m* = *FP*(*MPvo*) khi đó *vo* là vector khởi tạo của *P*.

**Ví dụ 3.1.1:** Xét chương trình *P* = {*p* ← *q*, *q* ← *p* ∧ *r*, *r* ← *s*, *s* ← } with *BP* = {*p*, *q*, *r*, *s*} vậy ma trận biểu diễn của nó *MP* ∈ 4×4 là một ma trận (vuông). Vector khởi tạo của *P* là *vo*= (0 0 0 1)T.   
ta có, *v*1 = θ(MP*v*0) = (0 0 1 1)T và *v*2 = θ(MP*v*1) = (0 0 1 1)T = *v*1.

Như vậy, vector *v*1 biểu diễn mô hình tối tiểu {*r*, *s*} của *P*.

Nghiên cứu [8] cũng giới thiệu việc tính điểm cố định của mô hình tối tiểu. Khác với nghiên cứu hiện nay, [8] cho một tập rỗng làm vector khởi tạo và tính điểm cố định.chúng ta bắt đầu bằng cách cho vector khởi tạo biểu diễn sự kiện, thay vì biểu diễn sự kiện rõ ràng trên ma trận. Điều này có tác dụng giảm các phần tử khác 0 trong các ma trận khi tính toán điểm cố định. [8] cho phép tồn tại ràng buộc “← *q*” trong chương trình trong khí các nghiên cứu hiện tại thì không. Các rang buộc được xử lý như một luật “*p* ← *q*, ¬*p*” trong phân 3.

### Non-SD program

Khi một chương trình xác định *P* chứa 2 luật: *r1*: *h* ← *b1* ∧ … ∧ *bm*  and *r2*: *h* ← *c1* ∧ … ∧ *cn*, *P* được biến đổi thành chương-trình-d *Q* sao cho:

*Q* = (*P* \ {*r*1, *r*2}) ∪ {*r*1’, *r*2’, *d*1}

Khi đó *r1*’: *h*1 ← *b1* ∧ … ∧ *bm*, *r2*’: *h2* ← *c1* ∧ … ∧ *cn* và *d1*: *h* ← *h1* ∨ *h2*.

ở đây, *h*1 và *h*2 là các biến mới với *r*1 và *r*2, tương ứng.

Tóm lại, chương trình khác-SD được biến đổi thành chương-trình-d như sau.

**Định nghĩa 3.1.6:** **(biến đổi)** Cho *P* là một chương trình xác định và *BP* là cơ sở Herband. Với mỗi *p* ∈ *P*, đặt *Pp* = { *r* | *r* ∈ *P* và *head*(*r*) = *p*} và *Rp* = { *r* | *r* ∈ *Pp* và | *Pp* | = *k* > 1}. Sau đó xác định *Sp* = {*pi* ← *body*(*r*) | *r* ∈ *Rp*, *i* = 1,…,*k*} và *Dp* = { *p* ← *p*1 ∨ … ∨ *pk* }. Xây dựng chương-trình-d:



Chúng ta có *P’* = *Q* ∪ *D* khi đó *Q* là một SD-program và *D* là một tập của d-rules.

Dễ dàng thấy được chương-trình-d *P’* có môt hình tối tiểu *M’* sao cho *M = M’* ∩ *BP* khi đó *M* là mô hình tối tiểu của *P*.

**Định nghĩa 3.1.7: (ma trân biểu diễn của chương-trình-d)** Cho *P’* là một chương-trình-d sao cho *P’* = *Q* ∪ *D* khi đó *Q* là một chương trình-SD và *D* là một tập của luật-d, và *BP’* = {*p*1*,…, pn’*} cơ sở Herband của *P’*. Thì *P’* được biểu diễn bằng ma trận *MP’* ∈ n’×n’ sao cho mỗi phần tử *aij* (1 *≤* *i*, *j* *≤* *n’*) trong *MP’:*



2. Trường hợp khác, mỗi luật trong *Q* được thể hiện trong định nghĩa. 2.4.

**Định lý 3.1.2**: Cho *P’* là một chương-trình-d va *MP’* ∈ *n*’×*n*’ là ma trân biểu diễn. Vậy *m* ∈ *n’* là vector biểu diễn mô hình tối tiểu của *P’* khi và chỉ khi *m* = *FP*(*MP’vo*) khi đó *vo* là ma trận khởi tạo của *P*’.

### Thuật toán tìm mô hình tối tiểu

Dựa trên định lý 3.1.2, chúng tôi xây dựng một thuật toán để tính mô hình tối tiểu của chương trình xác định *P*.

**Input:** chương trình xác định *P*, và cơ sở Herband *BP* = {*p*1*,…,pn*}.

**Output:** vector *u* biểu diễn mô hình tối tiểu của *P*.

**Thuật toán 3.1.1:**

|  |  |
| --- | --- |
| **Bước 1:** Biến đổi một chương trình xác định *P* thành chương-trình-d *P’* = *Q* ∪ *D với* *BP’* = {*p*1*,…,pn, pn*+1*,…,pn’*}, khi đó *Q* là một chương trình-SD và *D* là một tập luật-d.  **Bước 2:**  - khởi tạo ma trận *MP’ =* (*aij*)1≤ i,j ≤ n’ biểu diễn cho chương-trình-d *P’.*  *- K*hởi tạo vector khởi tạo *vo =* (*v1,…vn’*) của *P’.* | **Bước 3:** *tính toán mô hình tới tiểu của P’*  *v*:= *vo*  *u*:=*θ* (*MP’ v*)  while *u ≠ v* do  *v*:= *u;*  *u*:= *θ* (*MP’ v*)  end do; #while  return *u*[1…*n*]; |

**Ví dụ 3.1.2:** Xét *P* = {*p* ← *q*, *p* ← *r* ∧ *s*, *r* ← *s*, *s* ← } và *BP* = {*p*, *q*, *r*, *s*}. Sau đó *P* được biến đổi thành chương-trình-d *P’* = *Q* ∪ *D* với *BP’* = {*p, q, r ,s, t, u*}:

*Q* = { *t* ← *q*, *u* ← *r* ∧ *s, r* ← *s*, *s* ← }

 *D* = { *p* ← *t* ∨ *u* }.  .  
chúng ta có ma trận vuông *MP’* ∈ 6×6biểu diễn *P’*. Cho *v*0 = (0 0 0 1 0 0)T là vector biểu diễn sự kiện trong *P’*. Vậy, *v*1 = θ(MP’*v*0) = (0 0 1 1 0 0)T, *v*2 = θ(MP’*v*1) = (0 0 1 1 0 1)T, *v*3 = θ(MP’*v*2) = (1 0 1 1 0 1)T, *v*4 = θ(MP’*v*3) = (1 0 1 1 0 1)T = *v*3. Như vậy, *v*3 là vector biểu diễn mô hình tối tiểu của *P’*, và *v*3[1…4] là vector biểu diễn mô hình tối tiểu của *P*, {*p, r, s*}.

Trong thuật toán 2.1, độ phức tạp của *MP’v* là **O(*n’*2)**. Số lần để lặp lớn nhất của *MP’v* là (***n* + 1)** lần. Vì vậy,độ phức tạp của bước 3 là **O((*n* + 1)×*n’*2)** trong trường hợp xấu nhất.

### Tính bằng ma trận con

Phần này giới thiệu phương pháp để giảm độ phức tạp của việc tính toán *u* = *θ*(*MP’v*).

**Định nghĩa 3.1.8: (ma trận con biểu diễn của chương-trình-d)** Cho *P’* là một chương-trình-d sao cho *P’* = *Q* ∪ *D* khi đó *Q* là một chương trình-SD và *D* là tập hợp của các luật-d, và *BP’* = {*p*1*,…, pn’*} cơ sở Herband của *P’*. Sau đó *P’* được biểu diễn bằng ma trận *NP’* ∈ n’×n sao cho mỗi phần tử *bij* (1 *≤* *i ≤* *n’,* 1 *≤* *j* *≤* *n*) trong *NP’* tương đương với các phần tử *aij* (1 *≤* *i*, *j* *≤* *n’*) tương ứng trong *MP’* của định nghĩa.2.7. *NP’* được gọi là *ma trận con* của *P’*.

Lưu ý rằng kích thước *MP’* ∈ *n’*×*n’* của định nghĩa.2.7 bị giảm còn *NP’* ∈ *n’*×*n* trong định nghĩa.2.8.

**Định nghĩa 3.1.9: (hàm-ngưỡng-d)** Cho vector *v*= (*a*1*,…, an’*)T, định nghĩa một vector *w*= θ*D*(*v*) = (*w*1*,…, wn’*)T sao cho (i) *wi* = 1 (1 ≤ *i* ≤ *n’*) nếu *ai* ≥ 1, (ii) *wi* = 1 (1 ≤ *i* ≤ *n*) if ∃*j* *wj* = 1 (*n* + 1 ≤ *j* ≤ *n’*) và có một luật-d *d*∈*D* sao cho *head*(*d*) = *pi* và row*j*(*w*) ∈ *body*(*d*), và (iii) trường hợp khác, *wi* = 0.

θ*D* được gọi là một *hàm ngưỡng d* trên chương trình d *P’*. bằng định nghĩa của θ*D*, ∃θ*D*(*v*) = θ*D*(θ(*v*)).

**Định lý 3.1.3**: Cho *P* là một chương trình xác định với *BP* = {*p*1*,…,pn*}, và *P’* một biến đổi chương-trình-d với *BP’* = {*p*1*,…,pn, pn*+1*,…,pn’*}. Cho *NP’* ∈ *n’*×*n* là ma trận con của *P’*. cho vector *v* ∈ *n* biểu diễn mô hình *I* của *P*, cho *u* = θ*D*(*NP’v*) ∈ *n’*.

Vậy *u* là vector biểu diễn mô hình *J* của *P’* sao cho *J* ∩ *BP* = *TP*(*I*).   
 *Chứng minh:*

Cho khi đó với  (1 ≤ *k* ≤ *n’*)

+ Giả sử 

và *u* = θD(*M1*.*v*) = θD(*w*) = (*u*1*, …, un’*)T

• Chứng minh: J ∩ BP ⊆ TP(I)

Cho *uk* = 1 (1≤ *k* ≤ *n*) và *pk* = row*k*(*u*). Ta chứng minh: *pk* ∈ T*p*(*I*)

1. Trường hợp 1: Nếu *wk* = *uk* = 1: Vì *wk* = 1, nên 

Cho  với *bi* ≠ 0 (1 ≤ *m* ≤ *n*). Ta có:

*bi* = 1/*m* (1 ≤ *i* ≤ *m*) và 

Khi đó, tồn tại một luật: *pk* ← *pk1* ∧ … ∧ *pkm* thuộc P mà *pki* = col*j*(*M*1) khi *bi* = *akj* (1 ≤ *i* ≤ *m*) và tức là *pk* ∈ T*P*(*I*)

Ta chứng minh được *pk* ∈ T*P*(*I*) .

1. Trường hợp 2: nếu *uk* ≠ *wk*  thì *wk* = 0

Vì *wk* = 0, theo định nghĩa of θD :

∃ *ko*, *n* + 1 ≤ *ko* ≤ *n’*, *wko* = 1

và ∃*dko* ∈ D, row*ko*(*w*) = *pko* ∈ body(*dko*) và head(*dko*) = *pk*

. *dko*có dạng:

*pk* ← *pko1* ∨…∨ *pkoq* với *pko* ∈{*pko1*,…, *pkoq*} ⊆ BP’\BP

. *wko* = 1 (*n* ≤ *ko* ≤ *n’*): 

Cho  với *bi* ≠ 0 (1 ≤ *m* ≤ *n*). Ta có:

*bi* = 1/*m* (1 ≤ *i* ≤ *m*) và 

Khi đó tồn tại một luật: *pko* ← *pk1* ∧ … ∧ *pkm* thuộc Q

mà *pki* = col*j*(*M*1) khi *bi* = *akj* (1 ≤ *i* ≤ *m*) và 

Bằng cách chuyển đổi chương trình xác định P thành chương trình dương P’, ta có:

*pk* ← *pk*1 ∧ … ∧ *pkm* ∈ P và {*pk1*,…, *pkm*} ⊆ I

Do đó , *pk* ∈ Tp(*I*)

Ở cả 2 trường hợp 1 và trường hợp 2, ta có:

*pk*= row*k*(*u*) ∈ Tp(*I*) nếu *uk*= 1 (1 ≤ *k* ≤ *n*)

Khi đó J ∩ BP ⊆ TP(*I*)

• Chứng minh: TP(*I*) ⊆ J ∩ BP

Ta chứng minh: nếu *pk*∈ TP(*I*) thì *uk*= 1 (1 ≤ *k* ≤ *n*)

*pk*∈TP(*I*), vì vậy ∃{*pk1*,…, *pkm*} ⊆ *I* và *pk* ← *pk*1 ∧…∧ *pkm*∈ P (1 ≤ *m* ≤ *n*)

{*pk1*,…, *pkm*} ⊆ *I* , vì vậy *vkj* = 1 (1 ≤ *j* ≤ *m*)

+ Nếu *pk* ← *pk*1 ∧ … ∧ *pkm* ∈ Q thì:

*pki* ∈ BP ⇒ ∃ *j*, *pki* = col*j*(*M*1) (1 ≤ *i* ≤ *m*) và *akj*= 1/*m* (1 ≤ *j* ≤ *n*)

Do đó,  = 1 ⇒ *wk* = 1 = *uk*

+ Nếu *pk* ← *pk*1 ∧ … ∧ *pkm* ∉ Q thì:

∃*ko*, *n* + 1 ≤ *ko* ≤ *n’*, *pko* ← *pk1* ∧ … ∧ *pkm* ∈ Q

và *pk* ← *pko1* ∨…∨ *pkoq* ∈ D với *pko* ∈{*pko1*,…, *pkoq*}

. *pko* ← *pk1* ∧ … ∧ *pkm* ∈ Q thì:

*pki* ∈ BP ⇒ ∃ *j*, *pki* = col*j*(*M*1) (1 ≤ *i* ≤ *m*): *akoj*= 1/*m* (1 ≤ *j* ≤ *n*)

Vì vậy,  = 1 ⇒ *wko* = 1

Bởi vì *pk* ← *pko1* ∨…∨ *pkoq* ∈ D và row*ko*(*w*) = *pko* ∈{*pko1*,…, *pkoq*}

Nên *uk* = 1 (theo định nghĩa θD)

Ta có: nếu *pk*∈ TP(*I*) thì *uk*= 1 (1 ≤ *k* ≤ *n*) ⇒ TP(*I*) ⊆ J ∩ BP

Do đó: *J* ∩ BP = TP(*I*).

Cho ma trận *M1* ∈ n’×n, định nghĩa *v*1 = θD(*M1*.*v*0[1… *n*])

*v*2 = θD(*M1*.*v*1[1… *n*])

*vk*+1 = θD(*M1*. *vk*[1… *n*])

Cho một ma trận *NP’* ∈ *n’*×*n* và vector khởi tạo *v*o của *P'*, xác định *v*1 = θ*D*(*NP’v*0[1…*n*]) và *vk*+1 = θ*D*(*NP’* *vk*[1…*n*]) (*k* ≥ 1). Khi đó, *vk*+1 = *vk* với *k* ≥ 1. Khi *vk*+1 = *vk*, chúng ta viết *vk* = *FP*(*NP’v*0[1…*n*]). Ta thấy được *FP*(*NP’v*0[1…*n*]) biểu diễn mô hình tối tiểu của *P*.

Tóm lại, cho một chương-trình-d *P’*, giá trị *k* của *vk* = *FP*(*NP’v*0[1…*n*]) không lớn hơn giá trị *k* của *vk* = *FP*(*MPvo*) trong phần 3.1.2.

Bằng định lý 3.1.3, chúng ta có thể thay thế việc tính toán *u* = *θ* (*MP’v*) ở bước 3 của thuật toán 3.1.1 bằng *u* = θ*D*(*NP’v*[1…*n*]). Độ phức tạp của *NP’v*[1…*n*] là **O(*n’*×*n*)**,nó còn nhở hơn nhiều **O(*n’*2)** khi *n* << *n’*.

**Ví dụ 3.1.3:** Với chương-trình-d *P’* của ví dụ 3.1.2, chúng ta có ma trận con *NP’*∈ 6×4biểu diễn *P’*. Cho ma trận khởi tạo *v*0 = (0 0 0 1 0 0)T của *P’*, nó trở thành *v*1 = θD(*NP’v*0[1…4]) = (0 0 1 1 0 0)T, *v*2 = θD(*NP’v*1[1…4]) = (1 0 1 1 0 1)T, *v*3 = θD(*NP’v*2[1…4]) = (1 0 1 1 0 1)T = *v*2.

Vậy *v*2 là vector biểu diễn mô hình tối tiểu của *P’*, và *v*2[1…4] là vector biểu diễn mô hình tối tiểu { *p, r, s* } của *P.*

## Chương trình dạng chuẩn (Normal Program)

Chương trình dạng chuẩn được chuyển đổi thành chương trình dạng tuyển và sau đó giải mã chúng ở dạng ma trận cách sử dụng tensor bậc ba. Trong phần này, đầu tiên ta sẽ chuyển đổi chương trình dạng chuẩn thành chương trình xác định và sau đó giải mã chúng ở dạng ma trận như trong Mục 3.1.

### Tính toán mô hình ổn định

Một *chương trình dạng chuẩn* *P* là một tập hữu hạn các *luật* có dạng như sau:

 (3)

Với *h* và *bj* là biến mệnh đề. *P* được chuyển thành một chương trình xác định bằng cách viết luật ở bên trên lại thành như sau:

 . (4)

Với  là một mệnh đề mới tương ứng với *bi*. Ta có thể gọi *bi* là một mệnh đề và  là một mệnh đề phủ định

Cho một chương trình dạng chuẩn *P* và một tập *I* ⊆ *BP*, chương trình xác định sau khi biến đổi được ký hiệu là *P+*, được gọi là dạng *khẳng định*. Như trong định nghĩa 3.1.6, ta có thể biến đổi *P+* thành một d-program *P’*..

**Định lý 3.2.1 [12]:** Cho *P* là một chương trình dạng chuẩn. Khi đó, *I* là một mô hình ổn định của *P* khi và chỉ khi *I+* là mô hình tối tiểu của .

**Định nghĩa 3.2.1: (ma trận cho một chương trình dạng chuẩn)** Cho *P* là một chương trình dạng chuẩn với *BP* = {*p1,…,pk*}, và *P+* là dạng khẳng định của nó với . Cho *P’* là một d-program có được từ *P+* với *BP’* = {*p*1*,…,pk, pk*+1*,…,pm*, . Ta có {*q*m+1*,…,qn*} ⊆ {*p1,…,pk*}, và *P’* có *m* mệnh đề phủ định và (*n-m*) mệnh đề khẳng định.Khi đó, *P’* được biểu diễn bằng ma trận *MP’* ∈ *n*×*n* sao cho mọi phần tử *aij* (1 *≤* *i*, *j* *≤* *n*) trong *MP’:*

1. *aii* = 1 với *m* + 1 ≤ *i* ≤ *n*

2. *aij* = 0 với *m* + 1 ≤ *i* ≤ *n* và 1 ≤ *j* ≤ *n* với mọi *i* ≠ *j*

3. Ngược lại, *aij* (1 ≤ *i* ≤ *m*; 1 ≤ *j* ≤ *n*) có giá trị như trong định nghĩa 3.1.7.

Theo định nghĩa, mệnh đề phủ định được sinh ra trong *MP’* tương tự như sự kiện (fact). Ta có thể thấy *aii* = 1 cho  biễu diễn luật, nó có nghĩa đó là một “phỏng đoán” cho .

**Định nghĩa 3.2.2: (ma trận khởi tạo)** Cho *P* là một chương trình dạng chuẩn và *BP* = {*p1,…,pk*}, *P’* là biến đổi theo d-program của P (thông qua *P+*) và. *Ma trận khởi tạo* *Mo* ∈ *n* × *h* (1 ≤ *h* ≤ 2*n-m*) được định nghĩa như sau:

• Mỗi hàng của *Mo* tương ứng với mỗi phần tử của *BP’* ,với hàng*i*(*Mo*) = *pi* khi 1 ≤ *i* ≤ *m* và hàng*i*(*Mo*) = khi *m* +1 ≤ *i* ≤ *n*.

• *aij* = 1 (1 ≤ *i* ≤ *m*, 1 ≤ *j* ≤ *h*) khi và chỉ khi sự kiện *pi* ← thuộc *P*; ngược lại, *aij* = 0.

• *aij* = 0 (*m* + 1 ≤ *i* ≤ *n*, 1≤ *j* ≤*h*) khi và chỉ khi sự kiện *qi* ← thuộc *P*; ngược lại, *aij* có thể bằng 0 hoặc 1.

Cho *P* là một chương trình dạng chuẩn và *P’* là biến đổi theo d-program của P. Với mà trận chương trình *MP’* ∈ *n*×*n* và ma trận khởi tạo *Mo* ∈ *n* × *h*. Định nghĩa:

*M1* = *θ* (*MP’Mo*) và *Mk*+1 = *θ* (*MP’Mk*) (*k* ≥ 1).

Khi đó ,ta sẽ tìm được *Mk*+1 = *Mk* với một giá trị *k* ≥ 1 nào đó. Khi *Mk*+1 = *Mk*, ta viết *Mk* = *FP*(*MP’Mo*).

Giả sử *Mk* = *FP*(*MP’Mo*) (*k* ≥ 1). Cho *u*= (*a*1 *. . .am, am*+1*… an*)T là một vector cột của *Mk* sao cho *aj* = 1 (tương ứng với *aj* = 0) (*m* + 1 ≤ *j* ≤ *n*) khi và chỉ khi *ai* = 0 (tương ứng với *ai* = 1) với 1 ≤ *i* ≤ *m*, hàng*j*(*Mo*) = và hàng*i*(*Mo*) = *pi* = *qj*. Khi đó, ta có kết quả tiếp tiếp theo.

**Định lý 3.2.2**: *u* là vector cột biễu diễn tập *I* của *P’* khi và chỉ khi *I* ∩ *BP* là mô hình ổn định của *P*.

### Thuật toán để tính toán mô hình ổn định

Dữa trên định lý 3.2, ta có thuật toán tìm mô hình ổn định của một chương trình dạng chuẩn *P* như sau:

**Input:** chương trình dạng chuẩn *P* and tập cơ sở Herband *BP* = {*p*1*,…,pk*}

**Output:** Tập vector biểu diễn mô hình ổn định của *P*.

**Thuật toán 3.2.1:**

|  |  |
| --- | --- |
| **Bước 1:** Biến đổi chương trình dạng chuẩn *P* thành d-program *P’* (thông qua *P+*) với *BP’* = {*p*1*,…,pk,*  **Bước 2:**  - Tạo ma trận *MP’* ∈ *n* × *n*biểu diễn d-program *P’*.  *-* Tạo ma trận khởi tạo *Mo* ∈ *n*×*h* of *P’*.  **Bước 3:** *Tính điểm cố định FP(MP’Mo**)*  *M* := *Mo*  *U* :=*θ* (*MP’M*)  while *U ≠ M* do  *M*:= *U;*  *U*:= *θ* (*MP’M*);  end do; | **Step 4:** *Tìm mô hình ổn định của P*  result:={ };  for *i* from 1 to *h* do  *v***:**= (*a*1*,. . .am,am*+1*,… ,an*)T is *i*th-column of *M*  for *j* from *m* + 1 to *n* do  := row*j*(*M*);  for *i* from 1 to *m* do  if row*i*(*M*) = *qj* then  if *ai* + *aj* ≠ 1 then  break;  end for; #*i*  if *i* ≤ *m* then break;  end for; #*j*  if *j* ≤ *n* then break;  else result := result ∪ {*v*};  end for;  return result; |

**Ví dụ 3.2.1:** Xét *P* ={ *p* ← *q* ∧ ¬*r* ∧ *s*, *q* ← ¬*t* ∧ *q*, *q* ← *s*, *r* ← ¬*t*, *s* ←, *t*←} với *BP* = {*p*, *q*, *r*, *s, t*}. Đầu tiên, *P* được chuyển đổi thành dạng khằng định *P+* và một d-program *P’* như sau:

• *P+* = { *p* ← *q* ∧  ∧ *s*, *q* ←  ∧ *q*, *q* ← *s*, *r* ← , *s* ←, *t* ← }

• *P’*= *Q* ∪ *D* với: *Q* ={ *p* ← *q* ∧  ∧ *s*, *q1* ←  ∧ *q*, *q2* ←*s*, *r* ← , *s*←, *t*←}

*D* = { *q* ← *q1* ∨ *q2*}

Ta có ma trận *MP’* ∈ 9×9biễu diễn chương trình *P’* và ma trận khởi tạo *Mo* ∈ 9×2:



Khi đó *M1* = *θ* (*MP’Mo*), *M2* = *θ* (*MP’M1*), *M3* = *θ* (*MP’M2*) trở thành:



*M4* = *θ* (*MP’M3*) = *M*3 chính là điểm cố định.

Trong trường hợp này, vector cột *u* = (1 1 0 1 1 0 1 1 0)T thỏa mãn điều kiện *a*8 = 1 khi và chỉ khi *a*3 = 0 với hàng8(*u*) =  và hàng3(*u*) = *r*. Vector *u* biễu diễn tập {*p, q, s, t, q*2,} và {*p, q, s, t, q*2, } ∩ *BP* = {*p, q, s, t*} là mô hình ổn định của *P*.

Ở thuật toán 3.1, độ phức tạp của của phép toán *MP’M* là **O(*n*2×*h*)**. Số lần lặp của phép toán *MP’M* nhiều nhất là (***n* + 1)** . Do đó, đọ phức tạp ở Bước 3 là **O(**(***n* + 1) ×*n*2×*h*)** trong trường hợp xấu nhất.

### Tính toán bằng ma trận con

Chúng ta áp dụng kĩ thuật ma trận con ở định nghĩa.2.8vào ma trận biểu diễn cho chương trình dạng chuẩn. Nghĩa là, thay vì xét một ma trận biểu diễn *MP’* ∈ n×n của định nghĩa.3.1, chúng ta xét một ma trận con *NP’* ∈ n×n’  khi *n’=k+n-m.* Chú ý, *n’<< n* in general.

**Định nghĩa 3.2.4:** (**vector khởi tạo**) Cho *P* là một chương trình dạng chuẩn với *BP* = {*p1,…,pk*}, và

*P+* ở dạng khẳng định với *BP+* = {*p1,…,pk*, ,… } khi {*qk+*1*,…,qn*} ⊆ {*p1,…,pk*}.

thì tập các vector khởi tạo của một dạng phủ định *P+* được định nghĩa như sau:

cho *v*1 ∈ *k* là vector biểu diễn sự kiện trong *P* khi row*j*(*v*1) = *pj*.

xét *A* ⊆ *n* - *k* khi *A* = {(1 0 … 0)T, (1 1 … 0)T, …., (1 1 … 1)T } với card(*A*) = 2*n-k*, và *B* = {*v* ∈ *A* | ∃ *i* (1≤ *i* ≤ *n-k*) s.t. *v*[*i*] = 1 và ∃ *j* (1≤ *j* ≤ *k*) s.t. *v1*[*j*] = 1 và row*j*(*v1*) = *p* khi và chỉ khi row*i*(*v*) =  khi *v*1 biểu diễn sự kiện trong *P*}. đặt *v*2 ∈ *n-k* s.t. *v*2 ∈A\B. tập các vector khởi tạo *V* của *P+* là:



Theo định lý 2.3, nếu *v* ∈ *n* là vector khởi tạo mô hình *I* của *P+*, và *u* = θD(*NP’* *v*) ∈ *n’*, thì *u* là một vector biểu diễn mô hình *J* của *P’* và *J* ∩ *BP+* = *TP+*(*I*).

Vì lý do này, trong bước 3 của thuật toán 3.1, chúng ta có thể thay thế việc tính toán điểm cố định *FP*(*MP’Mo*) thành tính toán điểm cố định *FP*(*NP’v*o[1…*n*]) với vector khởi tạo *v*o∈*V*. độ phức tạp của việc tính toán *FP*(*MP’Mo*) là **O(**(***n* + 1)×*n*2×*h*).** Mặt khác, độ phức tạp của *NP’v*o[1…*n*] là **O(*n*×*n’*)**,khi số lần thực hiện phép nhân lớn nhất là (***n* + 1)** lần, chúng ta có |*V*| = *h*, vậy độ phức tạp bước 3 của thuật toán này là **O(**(***n* + 1) ×*n*×*n*’×*h*)** với *n’* << *n*.

**Ví dụ 3.2.2:** xét một chương trình dạng chuẩn *P* và một chương-trình-d *P’* của ví dụ 3.1.

 Chúng ta có ma trận con *NP’* ∈ 9×7biểu diễn *P’*:

• *v1* ∈ *5* biểu diễn sự kiện trong *P*, *v1* = (0 0 0 1 1)T



*V* = {(0 0 0 1 1 0 0)T, (0 0 0 1 1 1 0)T} với *h* =card(V)= 2

Tính điểm cố định *FP*(*NP’* *vo*) (*vo* ∈ *V*):

(i) Cho *vo* = (0 0 0 1 1 0 0)T: *v*1 = θD(*NP’vo*) = (0 1 0 1 1 0 0 0 1)T

*v*2 = θD(*NP’v*1[1…7]) = (0 1 0 1 1 0 0 0 1)T = *v*1

row3(*v*1) = *r* và row6(*v*1) =  then *v*1[3] + *v*1[6] = 0, vậy *v*1 không biểu diễn mô hình ổn định của *P*.

(ii) Cho *vo* = (0 0 0 1 1 1 0)T: *v*1 = θD(*NP’vo*) = (0 1 0 1 1 1 0 0 1)T, *v*2 = θD(*NP’v*1[1…7]) = (1 1 0 1 1 1 0 0 1)T, *v*3 = θD(*NP’v*1[1…7]) = (1 1 0 1 1 1 0 0 1)T = *v*2.

row3(*v*2) = *r* and row6(*v*2) =  then *v*2[3] + *v*2[6] = 1

row5(*v*2) = *t* and row7(*v*2) =  then *v*2[5] + *v*2[6] = 1

*v*2 biểu diễn tập {*p, q, s, t,* } và {*p, q, s, t,* } ∩ *BP* = {*p, q, s, t*} là mô hình ổn định

# KẾT QUẢ THỬ NGHIỆM

Trong chương này, chúng tôi sẽ tiến hành việc thực nghiệm các thuật toán tìm mô hình tối tiểu của một chương trình logic xác định và tập hợp của các mô hình ổn định một chương trình dạng horn và dạng chuẩn bằng phương pháp điều-kiện-MD ban đầu, phương pháp điều-kiện-SD và phương pháp ma trận con của chúng tôi. So sánh và đưa ra kết luận. Thử nghiệm được tiến hành trên hệ thống có cấu hình như sau:

* Operating system: Window 10.
* CPU: Intel® Core™ i5-8400 <4.0 GHz /14nm / Cores = 6 / Threads = 6 / Cache = 9 MB>, Memory 8GB, DDR-2400
* GPU: GeForce GTX1060 GDDR5 6GB
* Implementation language: Maple 2018, 64 bit [14].

## So sánh với chương trình logic dạng horn

Trong thử nghiệm này, cho giá trị *n* = *|BP*| là kích thước của tập cơ sở Herband *BP* và giá trị *m* = *|P*| là số lượng các luật trong *P*, các luật được tạo ngẫu nhiên như trong Bảng 1:

**Bảng 1.** Tỉ lệ của các luật trong *p* theo số lượng của các biến mệnh đề trong phần thân

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Số lượng các phần tử trong body | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Số lượng các luật (theo tỉ lệ %) | x, where x < n/3 | 4% | 4% | 10% | 40% | 35% | 4% | 2% | ~1% |

Trong mục này, chúng tôi sẽ tiến hành việc thực nghiệm và so sánh các thuật toán tìm mô hình tối tiểu của một chương trình logic xác định và thuật toán tìm các mô hình tối thiểu của một chương trình logic dạng tuyển bằng phương pháp các gốc dựa trên việc tính toán ma trận tensor, *TP*-operator, chương trình SD và tính toán ma trận con.

Dựa vào cặp giá trị (*n, m*), một chương trình xác định *P* sẽ được tạo ra một cách ngẫu nhiên. Sử dụng Bổ đề 2.2.1, ta chuyển đổi *P* thành một chương trình xác định *P’* thỏa mãn MD-điều kiện. Từ chương trình xác định P ban đầu, ta chuyển đổi P thành một chương-trình-d *P’* = *Q* ∪ *D* với *Q* là một SD-program và *D* là một tập d-rules.

Bảng 1 là kết quả của việc thử nghiệm thuật toán 2.2.1, 3.1.1 và 3.1.4 trên *P’* để tìm mô hình tối tiểu của một chương trình *P*. Có 2 bước quan trọng trong thuật toán 2.2.1, 3.1.1 và 3.1.4: Bước 1 để tạo ma trận *MP’* biểu diễn chương trình *P’*, và bước 3 để tìm điểm cố định. Ta so sánh 4 phương pháp:

* **(Phương pháp 1)**: tính *MP’**v* bằng thuật toán 2.2.1.
* **(Phương pháp 2)**: Sử dụng *TP*-operator trên chương trình *P*.
* **(Phương pháp 3):** tính toán đơn giản một chương trình *P’* bằng Thuật toán 3.1.1.
* **(Phương pháp 4):** sử dụng tính toán ma trận con trên một chương trình *P’* như ở Mục 3.1.4. Biểu đồ 1 so sánh thời gian tính toán điểm cố định trên chương trình xác định.

**BẢNG 2**.kết quả thử nghiệm trên chương trình xác định

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Data | n | m | **Phương pháp 1** | **Phương pháp 2** | **Phương pháp 3** | **Phương pháp 4** |
| 1 | 20 | 400 | 0.266 | 0.016 | 0.344 | 0.015 |
| 2 | 20 | 8000 | 87.656 | 0.125 | 53.547 | 0.25 |
| 3 | 50 | 1250 | 3.61 | 0.062 | 2.828 | 0.094 |
| 4 | 50 | 2500 | 13.187 | 0.11 | 12.828 | 0.25 |
| 5 | 50 | 12500 | 411.812 | 0.391 | 223.578 | 0.765 |
| 6 | 100 | 5000 | 35.938 | 0.281 | 27.203 | 0.625 |
| 7 | 100 | 10000 | 205.407 | 0.672 | 80.625 | 1.25 |
| 8 | 200 | 400 | 0.141 | 0 | 0.047 | 0.11 |
| 9 | 200 | 13300 | 708.718 | 1.359 | 259.313 | 2.937 |

Kết quả trên là tổng thời gian (giây) để tạo ma trận biểu diễn + thời gian tính toán điểm cố định

**Biểu đồ 1.** Kết quả thử nghiệm việc tính toán điểm cố định trên chương trình xác định

Theo kết quả thực nghiệm trên, ta có thể thấy: Phương pháp sử dụng điều-kiện-SD hiệu qua hơn gần gấp 2 lần so với phương pháp điều-kiên-MD và càng nhanh hơn khi dữ liệu càng lớn. Phương pháp Sử dụng *TP*-operator

## So sánh với chương trình logic dạng chuẩn

Trong thử nghiệm này, chương trình dạng tuyển *P* được tạo ngẫu nhiên như sau:

- Cho giá trị *n* = *|BP*| là kích thước của tập cơ sở Herband *BP*.

- Cho giá trị *m* = *|P*| là số lượng các luật trong *P*.

Về phần đầu (vế trái) của các luật:

BẢNG 3. tỉ lệ của các luật trong *p* dựa trên số lượng của các biến mệnh đề trong VẾ TRÁI (HEAD)

|  |  |  |  |
| --- | --- | --- | --- |
| Số lượng phần tử trong head | 3 | 2 | 1 |
| Số lượng luật | x với x ≤ 3 | y với y ≤ 9 | còn lại |

Do đó, ta có số chương trình dạng Horn có thể được tách từ *P* là.

Về vế phải (body) của luật: Tỷ lệ số lượng các luật dựa trên số lượng của các biến mệnh đề trong vế phải như Bảng 1.

Do đó, chương trình *P* có nhiều hơn 95% số luật chứa nhiều hơn một biến mệnh đề trong phần thân.

Dựa trên cặp giá trị (*n, m*), một chương trình dạng tuyển *P* sẽ được tạo ra một cách ngẫu nhiên. Ta chuyển đổi *P* thành chương trình *P’* thỏa mãn MD-điều kiện MD. Gọi *k* là số lượng của chương trình dạng Horn *SPh* được tách từ *P’*. Từ P ban đầu ta chuyển thành một chương trình dạng chuẩn. Sau đó, ta chuyển đổi *P* thành một d-program *P’* với *Neg* âm trông chương trình *P*.

Bảng 4 là kết quả của việc thử nghiệm thuật toán 2.3.1, 3.2.1 và 3.2.3 trên *P’* để tìm tập mô hình tối thiểu. Có 3 bước quan trọng trong thuật toán 2.3.1: Bước 2 để tạo một tensor *UP’* biểu diễn một chương trình dạng tuyển *P’*, bước 3.3 để tìm ma trận biểu diễn mô hình tối tiểu của các chương trình dạng Horn *SPh*, và bước 4 để tìm tập các vector biểu diễn mô hình tối thiểu của *P’*. Tương tự, Có 3 bước quan trọng ở Thuật toán 3.2.1: Bước 2 để tạo một ma trận biểu diễn biểu diễn một d-program *P’*, Bước 3 để tính toán điểm cố định và Bước 4 để tìm tập các vector dùng để biểu diễn mô hình ổn định. Ta so sánh 4 phương pháp sau:

* **(Phương pháp 1)**: Phương pháp tính *UP’**M* bằng thuật troán 2.3.1 cho bước 3.3 của chương trình dạng tuyển
* **(Phương pháp 2)**: Tính toán bằng *TP*-operator (sử dụng Định lý 3.2.1).
* **(Phương pháp 3):** **):** tính toán đơn giản trên một chương trình *P’* bằng Thuật toán 3.2.1.
* **(Phương pháp 4):** sử dụng tính toán ma trận con trên chương trình *P’* như ở Mục 3.2.3 của Bước 3 của chương trình dạng chuẩn

**Bảng 3.** Kết quả thử nghiệm của chương trình dạng chuẩn

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data | n | m | *k* | Phương pháp 1 | *Neg* | Phương pháp 2 | Phương pháp 3 | Phương pháp 4 |
| 1 | 20 | 40 | 1152 | 2.531 | 9 | 0.078 | 0.188 | 0.144 |
| 2 | 20 | 400 | 288 | 95.484 | 6 | 0.656 | 0.313 | 1.5 |
| 3 | 50 | 100 | 288 | 4.015 | 7 | 0.562 | 0.063 | 0.343 |
| 4 | 50 | 1250 | 288 | 1480.4543 | 6 | 3.797 | 9.969 | 6.219 |
| 5 | 100 | 200 | 294 | 36.515 | 10 | 1.61 | 5.938 | 5.703 |
| 6 | 100 | 5000 | N/A | >4000 | 7 | 42.234 | 56.766 | 27.36 |
| 7 | 100 | 10000 | N/A | >4000 | 7 | 78.234 | 102 | 33.078 |
| 8 | 200 | 400 | 1152 | 558.953 | 10 | 2.797 | 26.578 | 7.813 |
| 9 | 200 | 13300 | N/A | >4000 | 10 | 2937.125 | 2447.454 | 707.75 |

Kết quả trên là tổng thời gian (giây) để tạo ma trận biểu diễn + thời gian tính toán điểm cố định. Với *k* là số lượng của chương trình dạng Horn *SPh* được tách từ *P’* và *Neg* số phần tử âm trông chương trình *P*

**Biểu đồ 2.** Kết quả thử nghiệm việc tính toán điểm cố định trên chương trình dạng chuẩn

Phương pháp 1 tính *UP’**M* bằng thuật troán 2.3.1 cho kết quả châm nhất và chậm hơn các phương pháp còn lại ở tất cả các trường hợp. Phương pháp 2 tính toán bằng *TP*-operator cho kết quả tốt nhất ở các dữ liệu nhỏ nhưng đến các dữ liệu lớn cụ thể từ n=100 và m gấp 2 lần n thì phương pháp ma trận con cho kết quả tốt hơn rất nhiều, đặt biệt với n=200 và m=13300 phương pháp ma trận con cho kết quả nhanh gấp 4 lần so với tính toán bằng *TP*-operator.

# KẾT LUẬN

## Chủ đề cấp độ 2

Nội dung …………………

Nội dung………………….

### Chủ đề cấp độ 3

#### Chủ đề cấp độ 4

## Chủ đề cấp độ 2

TÀI LIỆU THAM KHẢO

Theo chuẩn IEEE

1. Trong [8], sự kiện được biểu diễn bằng cách “*pi* ← ” và được thể hiện trên ma trận bằng cách *aij* = 1 khi đó row*i*(*MP*)= *pi* and col*j*(*MP*) = . [↑](#footnote-ref-1)